**Classification Approach Description**

# **Methodology *(Replicability; Scalability; Interpretability)***

Please provide a detailed description of the methodology used for identifying the classification of the online job advertisement. The description should contain (1) the data processing steps, (2) the methods and models used, (3) references to the scientific papers/sources that present the methods and models used, and (4) the time it took to process the data set and classify the job advertisements.

Bear in mind that the workflow will be also evaluated based on the criteria for the Reusability and Innovativity Awards.

*This section will be evaluated for:*

*(1) the Replicability criterion: likeliness that the described approach can successfully reproduce the solution submitted by the team for the Accuracy award*

*(2) the Scalability criterion: amount of modification required for the approach to apply to similar datasets on a potentially larger scale*

*(3) the Interpretability criterion: the extent to which a human could understand and articulate the relationship between the approach’s predictors and its outcome; how well the logical reasoning behind the model which is making the prediction is developed (whether it is mathematically and/or technically sound*

|  |
| --- |
| Briefly, my solution translates all non-English job description and titles to English, followed by removing irrelevant text from the job descriptions using regex keyword search, as well as semantic search using a sentence-transformer i.e. Sentence-Bidirectional Encoder Representations from Transformers (Reimers, 2019). A semantic search is then conducted again, this time to highlight the top five ISCO classifications for each job description. A more computationally expensive cross-encoder is then used to determine the best ISCO classification of the five, thus classifying the job description.  The details of my solution are as such:   * **Data cleaning for translation and Translation**   + [Negligible time spent] Removal of “Notes” and “Some related occupations classified elsewhere:...” parts of ISCO descriptions. They appear at the end of some ISCO descriptions.     - Rationale: These snippets tend to point toward some other jobs. When performing semantic search, a ISCO description may falsely have high similarity with job descriptions because of text within these snippets.   + [Negligible time spent] Remove any special characters from job titles and description offhand which may confuse language detector and translator.   + [2-3 minutes] Language detection of job title and job description, using Lingua. Align with input format of Google Translate API (i.e. convert “SLOVENE” to “SLOVENIAN”)     - Rationale: Lingua allows us to specify the languages to be detected, which in this case would be all official EU languages. This ensures the outputs (detected languages) are consistent and prevents any downstream errors i.e. feeding wrong languages into the Google Translate API. Furthermore, the language of job description is prioritised over job titles, as the ISCO classification methodology is based on job descriptions rather than job titles. Nevertheless, the job title is still included as it could be informative still.   + [5 hrs] To combine job title and description e.g. “Job title. Job description...” (hereafter, to be indicated as just “job description”), split into sentences and send to Google Translate API to translate non-English job descriptions to English (~6-7k), using detected language from the previous step.     - Rationale: Translation of all job titles + descriptions to English might be a good idea, as English documents are heavily used to train embedding models. Furthermore, multilingual models may not have good performance for languages which are uncommon, which stems from how much data in that language was used to train the embedding models. This eliminates to possibility of poor performance stemming from the commonality of a language, which thereby affects the availability of data. Lastly, translation occurs on a sentence level to prevent Google Translate API from rejecting translations of very long documents.   + [Negligible time spent] Further removal of non-ASCII characters and convert all text to lower case.     - Rationale: These characters may add noise to embeddings. * **Removing noisy text from job description using regex and semantic search with sentence-transformer embedding models**   + [Negligible time spent] Regex cleaning to filter out sentences which are irrelevant to applying the ISCO classification, using common keyword search, as well as any short sentences (< 3 characters in length). If all text was removed in this step, it would be filled with the text available right before this step.     - Rationale: Removing noisy text is important to reduce token length to speed up embedding processes, in addition to “purifying” the semantic meaning of each job description. Intuitively, this should improve the results of semantic search. These noisy texts tend be regarding PDPA laws and regulations (e.g. “personal data”, “discrimination”, “privacy policy”) which are typical for job descriptions, or texts which often are part of job search websites (e.g. “click here”, “receive notification”, “job alert”), which are likely from data sources which were not scraped and cleaned properly.   + [2-3 hours] semantic search using very small (33.4M parameters) sentence-transfer embedding model (BAAI/bge-small-en-v1.5). The cosine similarity between each embedded sentence in job descriptions and the query ("Job titles, professions, tasks and skills.") was calculated, retaining sentences exceeding cosine similarity of 0.5. The first sentence, which is the job title, is always retained.     - Rationale: The motivation for this step is the same as that for the previous regex cleaning step. However, because not all permutations of noisy text could be covered by keywords, semantic search could be used to remove noisy sentences *within* each job description, hinging on semantic similarity. A very small embedding model is used, as there were many sentences for just 26k rows of job descriptions. Using a larger model would take considerably more time. Hence, this would allow the solution to scale better. A cosine similarity of 0.5 was used as the threshold for acceptance, based on manual reviews of sentences which were removed i.e. should remove minimal information important to applying ISCO classification. The query was designed such that it summarises the essence of ISCO classification. * **Classification reframed as a semantic search problem.**   + [2-3 hours] Convert ISCO description, and cleaned job descriptions to embeddings using a small (335M parameters) sentence-transformer embedding model (mixedbread-ai/mxbai-embed-large-v1).     - Rationale: embedding models can convert entire paragraphs into vectors which considers its semantic meaning as a whole (in contrast to word2vec and related models). This is required as the ISCO classification takes into account semantic meaning, and not just keyword searches, as per the ISCO guidelines. Furthermore, this is likely close to how a person thinks when trying to apply the ISCO classification.   + [Negligible time taken] Perform a semantic search (cosine similarity) calculating the cosine similarity of all ISCO description and job description embedded vectors. For each job description, to get the top 5 most similar ISCO description and codes, based on having the highest similarity scores.     - Rationale: Cross-encoders, used in the last step, produce better predictions for cosine similarity, but are more computationally expensive as text is fed through the entire transformer architecture (Reimers, 2019). Hence, it is pertinent to conduct a pre-filtering stage using an embedding model first to get the top five similar ISCO descriptions/codes.   + [7-8 hours] Of the five most similar ISCO description, find the top-most similar one to each job description using a small (278M) sentence-transformer cross-encoder model (jinaai/jina-reranker-v2-base-multilingual). For this stage, the job descriptions are “queries”, while the ISCO descriptions/codes are the “documents” to be retrieved.   Further notes  Any sentence-transformer model used has a version fixed in the scripts. This would ensure that model inferences are reproducible. However, if the machine used run the scripts is not a silicon chip Macbook and if a GPU is available (see 3.classification.ipynb), the "device" to be used for inference should be changed from "mps" to "cuda". Otherwise, change it to "cpu", although this would result in long runtimes for cross-encoder inference.  If a larger dataset is being used, the process of embedding models can be sped up in two ways:   1. being stricter with the semantic search cleaning step by using a higher similarity score threshold, at the cost of information loss and thus likely drop in accuracy. This would lead to shorter job descriptions, hence speeding up the embedding process. 2. using a CUDA-compatible GPU for inference, such as the Nvidia T4. The MPS in my M1 Macbook Air is faster than using its CPU, but is still much slower than a proper CUDA-compatible GPU. |

## **Architecture**

Please provide a description of the architecture of your approach. A diagram of the architecture is considered of additional value. Indicate what modifications would be required to apply the approach to similar datasets on a larger scale.

*This section will be evaluated for:*

1. *the Architecture criterion: evaluated based on its modules, their cohesion and their configurability; an architecture which is modular and includes clear connections between modules or components receives a higher score*

|  |
| --- |
| Steps highlighted in **light green** indicate steps which can be adjusted when applying the solution on a larger dataset. |

# **Hardware Specifications *(Replicability; Scalability; Interpretability)***

Please describe the hardware specifications of the machines that were used to run the methodology.

*This section will be evaluated for:*

*(1) the Replicability criterion*

*(2) the Scalability criterion*

*(3) the Interpretability criterion*

**Machine 1**

|  |  |
| --- | --- |
| CPUs | Apple M1 chip, 8-core CPU with 4 performance cores and 4 efficiency cores, 16 GB RAM (unified with GPU) [M1 Macbook Air, 2020] |
| GPUs | Apple M1 chip, 7-core GPU (unified RAM with CPU) [M1 Macbook Air, 2020] |
| TPUs | - |
| Disk space | Around 3GB, including space required by sentence-transformer embedding models and intermediate outputs of scripts |

# **Libraries *(Maintainability)***

Please provide the libraries used for approach, if any, as well as the links to these libraries, if available.

*This section will be evaluated for:*

*(1) the Maintainability and openness criterion: use of libraries which are regularly maintained will yield higher scores. (Examples include pytorch, tensorflow, scikit-learn, pandas, numpy, etc.) The use of libraries which are openly available will yield higher scores.*

|  |
| --- |
| Nltk (13.5k Github stars): https://www.nltk.org  Lingua (1.1k Github stars): https://github.com/pemistahl/lingua-rs  Deep\_translator (12 Github stars*\**): https://github.com/nidhaloff/deep-translator  Sentence-transformers (15k Github stars): https://sbert.net  Transformers (133k Github stars): https://github.com/huggingface/transformers  Numpy (27.7k Github stars): https://numpy.org  Tqdm (28.5k Github stars): https://tqdm.github.io  Pandas (43.4k Github stars): https://pandas.pydata.org  All these packages are open-sourced, although embedding models from transformers and sentence-transformers may not be so.  *\* Deep\_translator is essentially used to allow batch translation to Google Translate’s API. Any other package which can do the same could be used in place of this package, with minimal changes to scripts.* |

# **Open license *(Maintainability)***

Please provide the open license of the provided code, if any.

*This section will be evaluated for:*

*(1) the Maintainability and openness criterion: whether the approach is open and under an open licens*e

|  |
| --- |
| Jina CrossEncoder model (179 likes on HuggingFace): Creative Commons Attribution Non-Commercial 4.0 (usage for research is ok, see [here](https://huggingface.co/jinaai/jina-reranker-v2-base-multilingual))  Mxbai embedding model (505 likes on HuggingFace): Apache license 2.0  BGE embedding model (233 likes on HuggingFace): MIT License  Lingua: Apache license 2.0  NLTK: Apache license 2.0  deep\_translator: MIT License  sentence-transformers/ transformers: Apache license 2.0  All other packages are open-sourced and open license.  Any code written by me is open-sourced and open license. |

## **Similarities/differences to State-of-the-Art techniques *(Originality)***

Please provide a list of similarities and differences between the used methodology and to the state-of-the-art techniques.

*This section will be evaluated for:*

*(1) the Originality of the approach criterion: compare the approach used to the state-of-the-art; the extent to which the submission represents an improvement over these pre-existing approaches*

|  |
| --- |
| The basic structure of my solution uses semantic search, which relies on sentence-transformers, the current state-of-the-art for capturing the semantic meaning of text data, and representing them as vectors. Performing semantic search allows matching of long documents to one another.  There were two things I tried which I was not found in literature (to the best of my abilities), but appears to work reasonably well:   1. Removing irrelevant text data embedding within text document using a very small sentence transformer embedding model    1. State-of-the-art: Regex, TF-IDF filtering, fine-tuned T5/BART for classification, Named Entity Recognition (requires training).    2. Explanation:       1. As mentioned above, usually text data which is irrelevant to the context is removed using regex. However, regex does not work for text which is always presented in different ways in job descriptions e.g. description of company, working hours, salary etc. It is near impossible to find an exhaustive list of keywords which allow us to remove these snippets of text *within* a job description.       2. Using a very small sentence-transformer embedding model allows us to, firstly, consider the semantic meaning of each sentence and compare each sentence to a particular query that we want. This allows us to “pick” the relevant sentence from within the text, by simply adjusting the query. Hence, this could be a flexible method which can be applied in other domains as well, i.e. not just for ISCO classification.       3. Secondly, these sentence-transformers does not have to be fine-tuned with a labelled dataset, where acquiring and labelling a dataset would require even more time and resources.       4. Lastly, the computing resources required to perform this step is reasonable, given how it took a few hours on the CPU of a M1 Macbook Air, and would be much faster results if using a GPU. Applying this method resulted in a 9.2% reduction in mean word count, from 263 to 239. To multiply that by 26k rows, the total number of redundant tokens removed is substantial. This advantage grows with larger datasets.       5. This cleaning step was not optimised in the sense that I did not compare and contrast the results from using different queries, and picking the best one to use. The similarity score threshold could be optimised as well, as a higher threshold would lead to even harsher removal of redundant tokens, but it is likely it comes at the cost of increase false positives as well i.e. removal of ISCO-relevant information. 2. Using semantic search to perform zero-shot classification, where both query and documents could be long or short    1. State-of-the-art: word-embedding models (e.g. FastText, Doc2Vec), Sentence-BERT, large-language models (LLMs) for zero-shot classification, ColBERT, hybrid retrieval    2. Explanation: Based on the Massive Text Embedding Benchmark Leaderboard, sentence-transformers are used for specific tasks, such as retrieval, reranking, classification (small number of classes), and semantic search (Muennighoff, Tazi, Magne & Reimers, 2022). My solution attempts to combine all tasks into one, and stretches the classification task to consider many classes, 436 to be exact. Furthermore, based on the BEIR benchmark (which MTEB uses for the retrieval task) queries are usually short (1-2 sentences) while documents are long, for retrieval tasks (Thakur et al., 2021). In my solution, the job descriptions are framed as queries, which are often much longer than 1-2 sentences. The results show that long “queries” work reasonably well too, despite the models likely being trained with short queries, but long documents. Once again, this method has an advantage over the state-of-the-art, as the computing resources required are minimal e.g. as compared to LLMs and ColBERT. |

## **Contribution to scientific field *(Future orientation)***

Please describe how your submission contributed to the scientific field, what impact it could have and what could potentially be future work to improve the solution.

*This section will be evaluated for:*

*(1) the Future orientation and impact criterion: the potential effect of the approach used will be evaluated; this includes the scale of impact it has on the problem of the classification of job advertisements; the impact will be evaluated based on potential efficiency improvements and cost reductions.*

|  |
| --- |
| This solution has shown that pre-trained sentence-transformer models perform decently for the task of job classification, without any fine-tuning or training. This circumvent the needs to use the computational resources and time just to train a model which is specialised in classifying jobs (although the accuracy of such a model is likely to be much higher, especially if clean text data is used).  This method could be improved for scaling by using CUDA-compatible GPUs, which would speed up the inference steps in the pipeline a lot. Most time is taken up by the inference steps. An M1 Macbook air was probably too lightweight a machine for this method, which is why it took so long (~19 hours) to run one iteration of the pipeline. That being said, it is recognised that the Google Translate step is a bottleneck, and better hardware would not improve this.  Furthermore, speed (and possible accuracy) could be improved by optimising the removal of redundant text embedding in job descriptions, by making it as harsh as possible, without reducing accuracy. Shorter and more concise job descriptions should lead to better predictions and classification. Related to this, cleaning job descriptions using semantic search + sentence transformer embedding models seemed to work relatively well. A manual review of the removed sentences does show that irrelevant parts of the job description were removed. These components tend to be the description of the company, PDPA, and scraped web elements. This was slightly helpful in reducing the length of texts, which helps longer job descriptions fit more relevant information into embedding models with smaller context lengths. |

## **Lessons Learned *(Future orientation)***

Please state any lessons learned during the competition.

*This section will be evaluated for:*

*(1) the Future orientation and impact criterion: what were the lessons learnt during the competition, and what could potentially be future work to improve the solution.*

|  |
| --- |
| Data  Some job descriptions were so dirty and non-informative, only the job title was of any value e.g. job description only containing elements of websites, or description of an HR consultancy firm who posted the job. These job postings should not have been part of the classification exercise and scoring. An analysis of which website such job postings tend come from would be useful, as they should have been removed since it would be impossible to apply the ISCO framework on these job postings anyways.  Data processing  I should have been more careful with the cleaning of data coming from other languages, especially when removing non-ASCII characters. This might have affected the way the sentences were split, and thereby affecting all downstream processing e.g. removing noisy sentences, embedding model inference. It is the first thing I would want to improve, if there was more time.  Classification  Hybrid retrieval (sparse BM25 retrieval + dense sentence-transformer embedding) did not lead to better classifications than just using dense retrieval (just the dense sentence-transformer embedding). Using a higher ratio of BM25 performance in hybrid retrieval led to poorer performance. This suggests that the semantic capability of any methodology is essential for job classification. This was based on a self- derived test dataset (a sample of 2 job descriptions for each language).  In the last step, it was not explored what the actual sentence similarity between ISCO description and job descriptions were. A diagnosis of cases with low sentence similarity from the cross-encoder step would have better informed areas where the solution should improve in. Furthermore, this method could also be extended, and first classify any predictions which are strongly discriminant and/or have high similarity scores between ISCO descriptions and job descriptions. A separate method could be applied on job descriptions with low sentence similarity from the last step of my pipeline.  Interestingly, longer-context (~8k) and multilingual models such as bge-m3 and jina-embeddings-v2-base-en did not lead to improved semantic search performance on my self-made test set. Using mxbai-embed-large-v1 for the initial step of identifying the top 5 ISCO descriptions nearest to each job description provided better performance. This is despite only keeping the first 512 tokens of each document and query. Perhaps this suggests important information in a job description tends to be in front. This could have been enough for a decent comparison to be made. Furthermore, when looking at relatively small embedding models, mxbai-embed-large-v1 has high scores on MTEB for both the reranking and semantic textual similarity tasks. It had scores higher than bge-m3 and jina-embeddings-v2-base-en as well. A comparison of LCA after stratifying performance based on long (>512 tokens) and short query/documents could be made to support this. If performance is equivalent, it may suggest that the first 512 tokens is enough for this task.  Continuing from the previous point, performance on smaller embedding models and rerankers resulted to overall poorer performance as well. Hence, there is a sweet spot the job classification task, and it shows that potential off-the-shelf models have to be tested.  Using the dot product as a measure of similarity for the initial semantic search of top-k documents led to slightly worse performance, as compared to using cosine similarity (using mxbai-embed-large-v1). This makes sense, as the length of a job description should not affect its semantic similarity to the ISCO descriptions. |

# **Short description of the Team – area of expertise**

Please provide a description of the team, your area of expertise and contact information.

|  |
| --- |
| Zi Yu Toh, student at KU Leuven, following programme of MSc Statistics and Data Science, with deep interest in statistics, machine learning and NLP.  Phone number: +32 456 64 77 42  E-mail: tohziyu2@gmail.com |

**References**

International Labour Organization. (n.d.). International Standard Classification of Occupations (ISCO). ILOSTAT. <https://ilostat.ilo.org/methods/concepts-and-definitions/classification-occupation/>

Muennighoff, N., Tazi, N., Magne, L., & Reimers, N. (2022). MTEB: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

Reimers, N. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *arXiv preprint arXiv:1908.10084*.

Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., & Gurevych, I. (2021). Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*.